

Lessons Learned from
EARLY ADOPTERS
OF OBJECT TECHNOLOGY



Taligent™

LESSONS LEARNED FROM

EARLY ADOPTERS OF OBJECT TECHNOLOGY

A TALIGENT WHITE PAPER

Table of Contents

Taligent Background	3
Impetus For Studying Early Adopters of Object Technology	3
Case Study Background and Methodology	4
Categories of Learning	4
Participants and Application Profiles	5
Lessons Learned: The Voices of Experience	6
√ 1: Educate Management	7
√ 2: Recruit a High-Level Sponsor	7
√ 3: Set Expectations Appropriately	8
√ 4: Learn From Experienced Peers	8
√ 5: Recruit and Carefully Manage Outside Experts	8
√ 6: Target Developers Based on Personal Motivation	8
√ 7: Work in Small, Geographically Close Teams	9
√ 8: Plan for New Roles and Skills	9
√ 9: Target Object-Based Applications Judiciously	9
√ 10: Invest in Training	10
√ 11: Recognize That Analysis and Design Are Difficult	11
√ 12: Consider Building An Enterprise Model	11
√ 13: Choose an Object-Oriented Language Carefully	12
√ 14: Evolve Prototypes Into Production Systems	12
√ 15: Conduct Regular Design and Code Reviews	13
√ 16: Focus Early on Performance	13
√ 17: Position Reuse as a Long-Term Benefit	13
√ 18: Develop Organizational Structures to Promote Reuse	14
√ 19: Create Reuse Incentives	14
√ 20: Develop Object-Oriented Metrics	14
Epilogue	15
Implications For Taligent	15

Taligent Background

Taligent, Inc. is an independent software company founded in March 1992 by Apple Computer, Inc. and IBM Corporation. Uniting a multi-year internal Apple® project, codenamed “Pink,” with some of IBM’s® most advanced technologies, Taligent is developing an exciting new technology foundation that will usher in the next wave of computing.

Scheduled for a mid 1990s delivery, the new Taligent™ system software platform and integrated development environment will have far greater scope than a traditional desktop operating system. Built completely with object technology, this platform offers an open, extensible computing environment – rich in function – that will accelerate the pace of innovation by systems and software vendors throughout the industry.

Uniting an internal Apple project with some of IBM’s most advanced technologies, Taligent is developing an exciting new technology foundation that will usher in the next wave of computing.

Dedicated to improving end-user computing and enabling dramatic improvements in application development productivity, Taligent intends to generate wide support for its new platform. The company will license, market and support its technology worldwide to software companies, hardware and systems vendors, and customers. Partnerships and cooperation, within the industry and with customers, are key to Taligent’s strategy and critical to delivering meaningful end-user solutions.

Taligent is deeply committed to raising industry awareness and understanding about the benefits and uses of object technology. These insights draw on the experiences of those companies who, like Taligent, are most advanced in their use of technology. In this spirit, Taligent offers this study on early adopters of object technology.

Impetus For Studying Early Adopters of Object Technology

True or false? Scores of corporate developers are successfully implementing large-scale object-oriented applications which run important segments of their business.

Despite the perception held by many corporate developers that few mission-critical object-oriented applications exist, this study uncovered substantial evidence to the contrary. For early adopters, object technology is today’s reality, already transforming the business and technology landscapes of those companies willing to make the leap.

This desire to understand and share the experiences of these early adopters was the impetus for Taligent’s Object Technology Early Adopter Case Study, which is summarized in this paper. The insight and wisdom of these trailblazers will provide practical guidance for others looking to adopt object technology or improve their existing application development initiatives.

Taligent believes companies must be encouraged to share their object technology experiences – both successes and challenges – to give newcomers to object technology a better chance to succeed. As one study participant put it, “We can’t all be Kit Carsons blazing the trail!” The lessons to be learned from these early adopters are an invaluable resource: for the industry, to ensure that future object-oriented products better meet customer needs – and for corporate developers, to successfully nurture the growth of object technology in their enterprises.

This paper is written for the advanced object technology user as well as developers beginning their first projects. It assumes a basic understanding of object-oriented technology concepts. Actual experience implementing object-based applications, however, is not essential.

Case Study Background and Methodology

Taligent chose the companies for its case study from a pool of over 125 corporate developers and system integrators who have significant object-based application work underway. To qualify for inclusion, companies had to meet four criteria:

1. Have developed business-critical applications: Companies who object technology experience is confined to R&D labs, or whose object-oriented applications are not central to their business mission, were excluded.
2. Have object-oriented applications in production or about to go "live:" Taligent wanted to compare full life cycle experiences with the technology, particularly to understand the companies' experience with reuse and reduced maintenance.
3. Use "standard" object-oriented languages (C++, Smalltalk™, Eiffel™, etc.), tools or frameworks (MacApp™): A range of object-oriented products was included to avoid biasing results toward any one technology choice.
4. Have the potential to deploy object technology broadly within divisions or enterprises: This broad deployment intent demonstrates a strong commitment to the technology.

A cross-functional team of Taligent marketing and engineering staff conducted face-to-face interviews over a four-month period with more than 100 senior IS and business executives, project managers, evangelists and development teams. Unlike traditional studies based on telephone or mail surveys, these live interviews allowed Taligent to explore the subject areas which follow with unprecedented breadth and depth.

Study Participants



American Airlines



American Express



Andersen Consulting



Becton Dickinson



British Airways



CIGNA



CSX Technology



Eastman Kodak



EDS



Federal Express



ICI



Liberty Mutual



Texas Instruments

Categories of Learning

With each participating company, Taligent explored ten areas that mirror the full life cycle of an object-oriented development project:

- Initial reasons for adopting object technology

- Approaches for getting started
- Methodologies used, including enterprise modeling, analysis and design
- Selection of target applications
- Language choice
- Training approaches
- Development team structures
- Metrics
- Benefits to date
- Experiences with reuse

Most important, companies agreed to share key “lessons learned” from their object technology experiences, to provide practical examples and perspective to other companies considering object technology for application development. These principles and recommendations form the core of this paper.

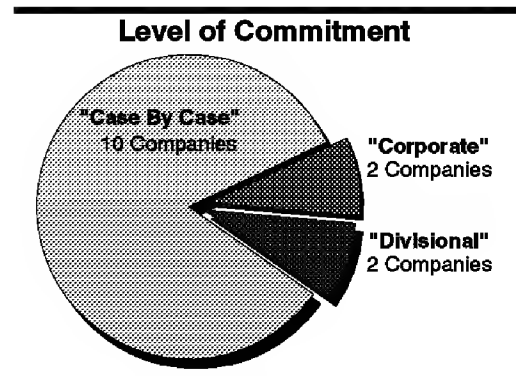
Participants and Application Profiles

To provide a context for the learnings which follow, Taligent gathered descriptions of the participants’ object-related experiences. The following section will give readers a feel for the diverse projects, technology choices and approaches found in these companies. Are they all early adopters? Yes! Are their experiences identical? Absolutely not!

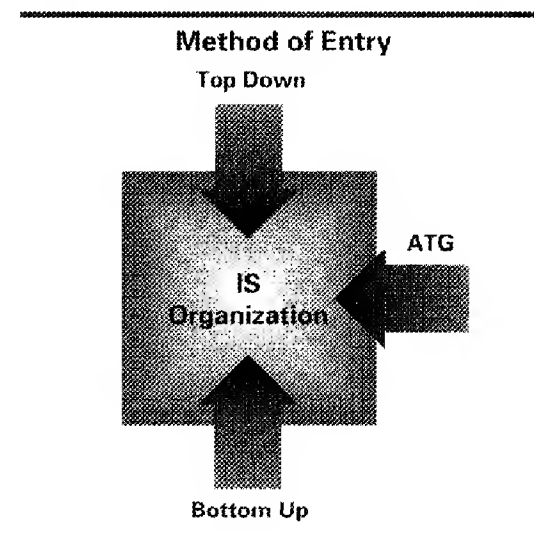
Reasons for Adoption: Not surprisingly, rapid application development and reuse were the top anticipated benefits which drove the adoption of object technology. Closely linked to reuse was the expectation that developers would be able to reduce the maintenance burden and more flexibly meet the needs of a rapidly changing business.

A range of other business and technology-related reasons for adoption were cited. Key business drivers were faster time-to-market, the ability to easily model the business, and fit with re-engineering initiatives. Other major technology benefits included rapid prototyping, easier GUI development, fit with

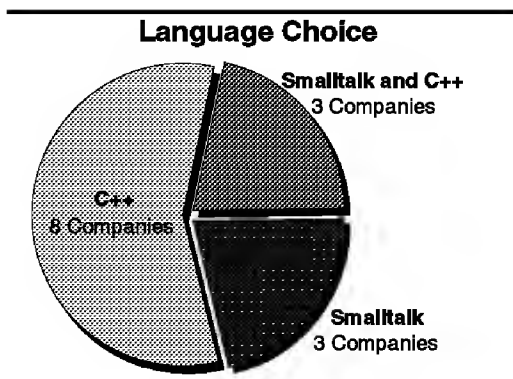
client/server and the belief that object technology offers a promising new foundation for the next generation of applications.



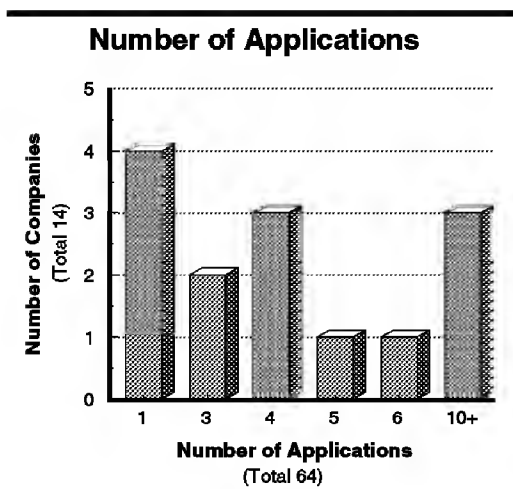
Corporate Commitment: All participants were clearly committed to object-based development, although to varying degrees. That four of the companies have made either corporate or divisional commitments may seem surprising. But those companies have bold visions of how object technology will help transform their businesses, and aggressiveness with new technologies has rewarded them in the past.



Method of entry: Historically, new technologies have entered corporate development organizations “from the bottom-up,” often fueled by the fervor of a technology evangelist. But object technology made its way into these companies by various routes, including traditional bottom-up approaches, adoption by advanced technology groups, and management-driven “top-down” decisions.



Language choice: While C++ was the dominant language choice, several companies strongly advocated Smalltalk. Companies selected C++ primarily because of higher performance, the perception that it will become the *de facto* standard object language, and the hope that C programmers could learn it easily. Developers who chose Smalltalk cited its ease of learning and strict enforcement of the object paradigm.



Number of applications: Altogether, 64 object-based applications were studied. Applications per company ranged from one to twelve, with the majority of participants engaged in multiple projects. While aggressiveness with the technology accounted for some of the disparity in projects per company, timing was the major determinant. The earlier object technology was adopted, the more applications a company was likely to have. Not surprisingly, most were planning additional object-oriented projects.

Application types: Participants described a remarkably wide range of applications, suggesting that object technology is applicable throughout their enterprises. Their applications addressed the following areas:

- Logistics, yield management and resource allocation
- Factory automation and process control
- Customer service
- Materials management
- Imaging workstation
- Medical/lab research and analysis
- Order management and billing
- Other applications including infrastructure layers, financial services, trading, motion control and document management

Application size: Applications ranged in size from 10 to 2,200 self-created classes. Nearly 60 percent of these projects had 200 or more classes, suggesting that, contrary to the perception of many corporate developers, companies are developing large-scale object-oriented applications.

Team size: Development teams ranged in size from 1 person to 30, with most averaging three to ten members. This suggests an emerging trend toward smaller teams.

Project length: While projects varied in length, over half were developed in under a year. These compact cycles reflect the ability to develop applications rapidly using object technology, as well as a fundamental shift from long-term, monolithic development to quicker delivery of pieces of functionality.

Lessons Learned: The Voices of Experience

For the Taligent team, the best part of the interviews came when participants were asked to reflect on key learnings – what worked well and what, in retrospect, they wish they had done differently in adopting object technology. Their feedback was analyzed and synthesized into the 20 key lessons presented here.

Taligent urges companies about to embark upon object technology initiatives to use these recommendations as guiding principles for their projects. They offer valuable advice for companies new to object technology or dissatisfied with their current application development practices.

Think of these recommendations as a checklist, a “recipe for success.” Their sequence follows the natural life cycle of a development project: from winning management support and sponsorship, to identifying and designing the target application, through project implementation and measurement of results.

Participants were asked to reflect on key learnings – what worked well and what they wish they had done differently in adopting object technology.

The reader will find cultural, organizational, technical and business-related ideas outlined here – and, often, crisply articulated in the participants’ own words. While many guidelines focus on issues unique to object-related projects, some apply equally to the introduction of any new technology, and others simply reflect good software engineering and business practices.

A final thought. Many of these lessons were learned the hard way, by trial and error, because these companies were navigating the uncharted waters of a very young technology. Their reward? A significant impact on the business which far outweighed the challenges of early adoption.

Those who follow these pioneers and benefit from their predecessors’ experience should find their own voyages into object technology easier to navigate. So, on to the lessons.

√ 1. **Educate Management**

Managers have difficulty supporting things they don’t understand. Simple, but true. Enthusiasts must remember that object

technology is truly a paradigm shift, accompanied by new concepts and terminology. Unlike “new” technologies that are extensions of old approaches, object technology “stands everything on its head.”

As one IS manager commented, “My executives are used to grasping new technology well enough to make decisions after a half-day seminar, and there’s no way you can comprehend object technology in that short a time.” Consequently, special courses for senior managers are critical to gaining their support.

Find business-oriented ways of selling object technology to executives. As one project manager astutely noted, “Object-oriented jargon like encapsulation, inheritance and polymorphism is meaningless to them, but concepts such as more flexible systems and reduced maintenance get their attention.” Participants noted that it’s easy to fall into the trap of selling object technology itself rather than the business benefits.

Be aware that there will be cultural barriers and organizational resistance to such a major technology change. These can be overcome only with senior management understanding and support. Virtually every company surveyed made this statement, conceding that such resistance remains an obstacle to success.

√ 2. **Recruit a High-Level Sponsor**

As with many new technologies, companies with high-level sponsors were more successful in securing the resources and management patience necessary to see a project through to completion. The sponsors receiving the highest marks were either technology-literate or keenly interested in understanding the impact of applying object technology to the business. The recommendation: Secure executive sponsorship early in the process, and ideally, find a sponsor who views technology as a strategic asset.

✓ **3. Set Expectations Appropriately**

Most companies' projects were significantly more difficult, expensive, and time-consuming than anticipated. This should not be surprising. For the first project, take care to set conservative expectations with end-users and executives. Build in a cushion, in terms of both time and budget. Sell the long-term vision, but make it clear that expectations of immediate payback on investment are unrealistic. Position shortened delivery cycles only for subsequent projects, after initial projects have established a solid base of experience.

✓ **4. Learn From Experienced Peers**

Don't reinvent the wheel. Most companies did not speak with object-experienced corporate peers before launching their own first projects. Without exception, they wish they had.

Participants offer this advice in retrospect: Identify companies with medium-to-large-scale, business-critical object technology projects, and learn from their successes and challenges. Attend object-oriented conferences featuring "user" speakers, and ask vendors and associations for introductions to object-experienced peers. Also, recognize that the experiences of corporate developers – rather than those of commercial developers – are the best barometer of success, because of the similar information technology environments and business models.

✓ **5. Recruit and Carefully Manage Outside Experts**

"Don't try to do it all yourself – get outside help, especially the first time out of the box," advised one evangelist. At a minimum, outside expertise is recommended for the critical areas of analysis and design, and training.

In seeking outside expertise, companies agreed, make sure to identify vendors and consultants with successful track records in large-scale object technology implementation. They observed that, as the object technology "frenzy" heats up, a lot of "experts" are freely throwing around object rhetoric. Said one development manager, "You've got to weed through those who talk a good game to find the ones who actually have the experience."

"Don't try to do it all yourself – get outside help, especially the first time out of the box."

Once the right consultant is engaged, make sure the internal development team "owns" the initiative. Develop a plan to ensure skills transfer from outside consultants to internal staff. As one manager ruefully noted, "Consultants are often expensive, and can hold you hostage if you don't begin knowledge transfer on day one."

✓ **6. Target Developers Based on Personal Motivation**

Worried that existing programming staff will have to be replaced when the move is made to object technology? Not so! Companies found that programming background – whether COBOL, Pascal, FORTRAN or C – is not a significant indicator of the ability to become a proficient object-oriented developer. Personality and intellectual curiosity are key. The aggressive and adventurous types will have the right attitude to attack an object technology project.

To ensure strong teams for their initial work, several companies posted object technology projects internal and assembled teams from the best-of-breed developers who applied. The result? Senior, experienced people got involved early and became mentors and coaches on subsequent projects.

√ 7. **Work in Small, Geographically Close Teams**

Given the current state of the technology, companies advised that five appears to be an ideal development team size. If large projects mandate bigger teams, break the application into subsystems and assign them to correspondingly small teams. Avoid geographically dispersed teams; they make communication and project coordination much more difficult. The best results were achieved when team members were sitting elbow to elbow.

Small, geographically proximate teams are especially important in object-oriented development. Given the lack of good tools for browsing and identifying objects for reuse, team members must communicate closely to foster reuse. “With objects, the days of a programmer hiding in the corner and coding by himself are over,” remarked one developer.

√ 8. **Plan for New Roles and Skills**

All of the companies stressed that object-based projects require new skills and management styles. The critical new roles to consider for object development are:

- **Evangelists** – to act as mentors, champions and object technology educators within the organization
- **Object analysts and designers** – to bring modeling discipline and methodology to the project
- **Architects** – to take ownership of the object model, and bring structure and consistency to the class hierarchies
- **Object librarians or a team ownership mechanism** – to manage class libraries and promote reuse across project teams

- **Object “consciences”** – to ensure that a rigorous object-oriented approach is applied
- **Object assemblers** – an emerging role, responsible for assembling pre-built objects from corporate class libraries into applications.

Companies warned that developers typically adapt more easily to these new roles than do the project managers and executives faced with creating and overseeing new organizational dynamics.

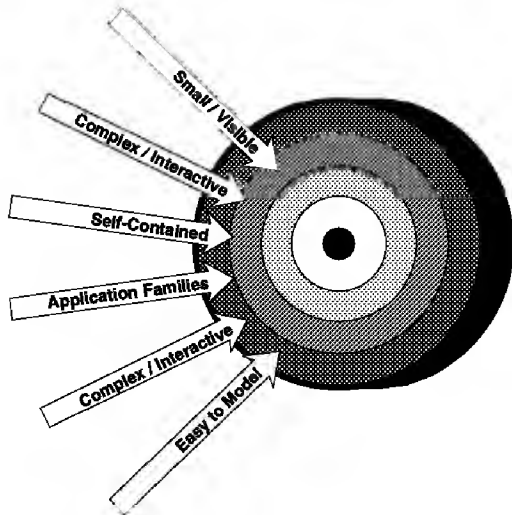
√ 9. **Target Object-Based Applications Judiciously**

What characterizes a good object-oriented application target? Companies identified several categories of applications that they believe most fully leverage the power of object technology.

Small, business-critical applications: While it is critical to identify important, visible projects that showcase the technology, don’t choose “bet the company” applications for initial object-oriented efforts. Start with smaller business-critical applications – meaningful “chunks” that can be delivered quickly – to increase the likelihood of success. But don’t select too-small or insignificant applications; ensure a valid payoff for the initial investment of time and money.

Complex, interactive applications: Many participants discovered that complex, highly interactive applications such as customer service or logistics management are a good fit for object technology. “If the application is too simple,” said one project leader, “you aren’t taking advantage of the technology’s ability to handle a high degree of complexity.” Object technology allowed several companies to model highly complex business problems which had eluded systems solutions in the past.

Application Targeting



Self-contained applications: Companies suggest identifying self-contained stand-alone applications, to avoid the pitfalls of dealing with legacy systems and interoperability issues. Great in theory...but most were unable to follow this suggestion in practice. Another recommendation was to target applications that are new or being completely redone, because object-oriented enhancements to traditional applications are currently difficult to implement.

Application families: Reuse is easier to achieve among applications “families,” or groups of applications sharing common functionality. Initially developing a set of common objects that can be used across multiple, related applications was viewed as a viable strategy for increasing developer productivity. For companies that deliver software-based products and need to customize solutions for niche markets, the ability to “copy and tweak” applications was significant.

Iterative development: Applications that require multiple rounds of user input, or whose requirements are difficult to “lock down,” are excellent object technology targets because of the ability to do iterative development. As one end-user liaison noted, “Object development is great for applications where you constantly have to pick their brains.”

Easy to model: Certain types of applications, which entail physical objects, lend themselves especially well to object-oriented implementation. For example, manufacturing and plant floor applications – which involve manipulation of physical components such as conveyor belts and automation devices – are good object technology targets because they are relatively easy to model.

One last message: Don’t tackle applications or systems that require too many changes at once. In several companies the simultaneous adoption of new hardware platform, new operating system, new development language and new development approach had clearly overwhelmed their people.

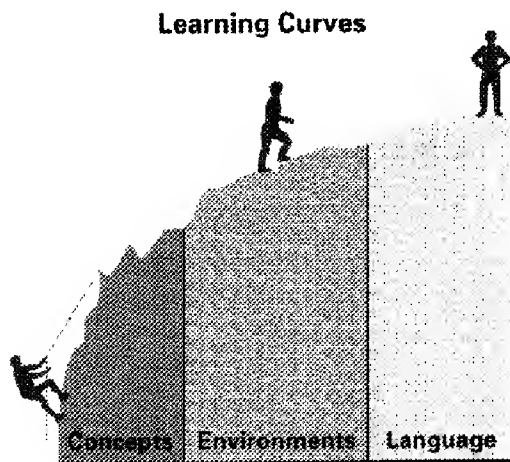
√ 10. Invest in Training

The need to invest in training was one of the most important lessons cited by these companies. Several participants noted that they made twice the training investment versus conventional programming methods. Investment aside, total immersion is essential. Said one programmer, “This is not the kind of technology you can learn while you’re working on an application development project at the same time.”

Every company emphasized the importance of adequate budgeting for outside trainers. Eight weeks or more of object-oriented training was not uncommon. Typically, top developers attend classes at nationally recognized training firms, then return to train their junior people – a kind of “train the trainer” approach. This method is often supplemented by engaging local consultants to design customized training tailored to the pilot project. Several companies have developed in-house training expertise to reduce reliance on outside firms.

A word from the wise. After completing their early projects, participants recognize three distinct learning curves, each with different challenges and requiring different levels of training:

- **Concepts:** Fully grasping the object paradigm – including concepts, analysis and design – is the hardest and longest learning task. “Learning object concepts will definitely mess with your head,” confirms one participant.
- **Environments:** Object development environments, including libraries and frameworks, are usually a moderate challenge.
- **Language:** Object-oriented languages are less of a problem for most, especially among experienced developers.



√ 11. Recognize That Analysis and Design Are Difficult

First and foremost, seek good outside consulting help for initial analysis and design. This is especially crucial for large, complex projects involving many programmers, or where rapid staff turnover is expected during the project. Roughly a third of the group – whether for lack of time, or because they considered their projects too small to warrant it – did not do analysis and design at this stage. Most regretted the decision.

With a virtual sea of object development methodologies being promoted in the marketplace, none has yet emerged as clearly superior or dominant – a fact these companies

found unsettling. Their advice, for now, is to pick the analysis and design methodology most relevant to your project, even if it is self-created, and stick with it to gain experience through the duration of the project. A consistent methodology will help ensure development of consistent classes and facilitate interoperability. And be prepared for the lack of robust, integrated design and analysis tools – for today, they’re almost non-existent.

√ 12. Consider Building An Enterprise Model

Enterprise models – working models of a company’s enterprise-wide business processes – helped a number of companies identify common sets of objects, which they focused on developing first. This foundation facilitated reuse of those common objects, thereby increasing developer productivity, reducing test and debug time, and resulting in higher-quality applications.

Make sure to validate the enterprise model with end-users – it cannot be an IS-only initiative. Some companies found it helpful to run a series of simulations to verify that critical business processes had been captured and the model was applicable to all segments of the business.

Here’s the rub. An enterprise model often calls for a centralized, well funded group to pay for and maintain the model. This requirement can be problematic, especially in more decentralized organizations. Although an enterprise model benefits individual business units, they are rarely willing – or able – to pay for it, as their applications are likely to leverage only a small portion of the model.

Nonetheless, companies who started by developing enterprise models gained a much clearer sense of direction for their projects, and more easily established a structured development approach. All in all, enterprise models should be an important consideration for companies adopting object technology.

✓ 13. Choose an Object-Oriented Language Carefully

The vast majority of participants were using C++ for production applications, driven largely by its better performance and the perception that it will become the *de facto* commercial standard. Still, there was stronger support than expected for Smalltalk, with a number of developers citing its pure approach to objects as a significant benefit.

Typically, companies did not rigorously evaluate available languages before choosing. The choice of language, in fact, was more often a matter of identifying the development environment and tools that best met their needs.

The lessons in this area boil down to this: Current languages all have strengths and weaknesses, but thorough analysis and evaluation increase the odds of a suitable choice. A number of other factors should weigh in the selection: the preference of the in-house evangelist, prior experience by team members, consultant recommendations, choice of

platform, and availability of development tools. Don't allow language choice to become a "religious" or "turf" issue – settle the question up front and move on.

Be aware that companies who use several object languages are having trouble developing corporate-wide class libraries because of interoperability challenges. One development manager admitted, "Our Smalltalk classes don't always incorporate smoothly with each other, and never with our C++ classes." Because of this obstacle, most of these companies have standardized on one language.

✓ 14. Evolve Prototypes Into Production Systems

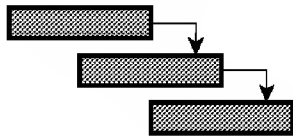
The old procedural world: A developer first prototypes an application, then discards that code and re-codes the production system. The new object world: Developers use an iterative approach, reusing or evolving prototypes into production systems. "Seamless" is how one developer described the iterative process.

C++ and Smalltalk

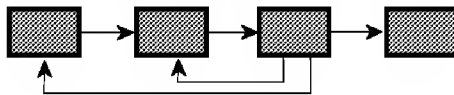
	Perceived Strengths	Perceived Weaknesses
C++	<ul style="list-style-type: none"> <input type="checkbox"/> Better performance <input type="checkbox"/> Perceived as emerging standard <input type="checkbox"/> Perceived to be similar to C <input type="checkbox"/> Good platform integration <input type="checkbox"/> Good cross-platform development <input type="checkbox"/> Allows access to low-level system features 	<ul style="list-style-type: none"> <input type="checkbox"/> Easier to fall out of OO paradigm <input type="checkbox"/> Difficult to learn; complicated syntax <input type="checkbox"/> Hybrid language <input type="checkbox"/> Few development environments <input type="checkbox"/> Lack of supplied class libraries
Smalltalk	<ul style="list-style-type: none"> <input type="checkbox"/> Easy to learn <input type="checkbox"/> Enforces object paradigm <input type="checkbox"/> More dynamic <input type="checkbox"/> Integrated class libraries <input type="checkbox"/> Availability of third party libraries <input type="checkbox"/> Platform independence and portability 	<ul style="list-style-type: none"> <input type="checkbox"/> Environment and applications integrated <input type="checkbox"/> Questionable real-time performance

Under the iterative model, applications can be developed side-by-side with the end-users in an ongoing code-and-test process. The days of gathering requirements, going off to code the application, and returning months – sometimes years – later with an application that no longer fits end-user needs, are over. Building an application iteratively also facilitates extending the system and adding value after the application is delivered.

Procedural Approach



Iterative Approach



The message to newcomers? Because object technology allows prototypes to be evolved into production systems, it makes little sense to prototype in one language and implement in another. The experiences of the few companies prototyping in Smalltalk and implementing in C++ bear out this lesson. They were unable to leverage the class libraries developed for the prototype when building the production system.

✓ **15. Conduct Regular Design and Code Reviews**

Regular design and code reviews enforce consistency among classes and help ensure that the analysis and modeling are solid before implementation proceeds too far. Establish this discipline early, and hold reviews at frequent intervals. Peer reviews also encourage reuse, because interactive communication in a review session makes developers more aware of the classes their fellow team members are developing.

✓ **16. Focus Early on Performance**

Because of the highly interdependent nature of their object-based systems, most companies wished they had anticipated and focused on system performance issues earlier in the development cycle. The failure to do so is likely a carryover from traditional programming methods, which typically leave performance tweaking to the end of a project. Don't wait!

✓ **17. Position Reuse as a Long-Term Benefit**

Reuse was the primary motivation for adopting object technology in most companies. However, the participants' experience indicates that reuse is not likely until the first couple of projects have been completed. In the words of one evangelist, "Reuse is definitely not the *initial* big win." To maintain credibility, don't sell object technology to management by promising them immediate reuse benefits. Position reuse as a long-term strategic asset.

What immediate benefits can you sell? Focus on increased flexibility, rapidly adapting applications to support a changing business environment, and lower maintenance requirements – benefits that often surface before reuse. Given that maintenance costs often far outweigh development costs for a typical application, reducing the maintenance burden through object technology may prove as valuable in the long run as increased developer productivity.

Don't sell object technology to management by promising immediate reuse benefits.

Think broadly about reuse, in terms of analysis and design as well as code. The insight? Design reuse may well precede code reuse and is potentially more valuable. Also, to accelerate the realization of reuse, look for

families of related applications with common functionality. Widely disparate applications with no business synergy make reuse difficult to achieve.

✓ 18. Develop Organizational Structures to Promote Reuse

Given the immaturity of today's object-oriented products, companies emphasized that the technology alone is not sufficient to ensure widespread reuse. Rather, reuse also requires sound management and organizational practices. "Reuse doesn't just happen all by itself," noted one development manager.

Reuse is easier to achieve if programmers have immediate access to the company's object resources. Object warehouses and libraries should be established – electronically, but also in hard copy for design and analysis documents. Without a central repository, isolated pockets of classes will emerge.

Reuse requires sound management and organizational practices.

Consider naming an object librarian to "productize" and maintain common objects and promote their reuse across development teams. Productizing entails the rigorous evaluation of all objects coming into the library – assessing whether they are well tested, documented, reusable – and assigning each object a name that communicates its purpose to developers. The librarian should also serve as a consultant, recommending object resources to those beginning new projects. The goal? To help developers find and use objects in less time than it would take to rewrite them.

Alternatively, consider having development teams "own" their domain-specific sets of classes. Ownership responsibility involves managing library development, class integrity and enhancements, and promoting reuse

across the company. This approach will require new skills, as distinctions between object developers and maintainers become less regimented.

The solutions described here are personnel-oriented, largely because current technologies – such as robust browser tools – to support object management and reuse objectives are not yet widely available. For the time being, the librarian concept was the winner hands down.

Finally, managers must realize that object-oriented projects require longer analysis and design phases and shorter coding phases. Managers who hold development teams to old, procedural schedules will short-circuit the upfront steps required to achieve reuse.

✓ 19. Create Reuse Incentives

Managers must create incentives for teams to reuse objects rather than write new code. For example, it makes sense in an object environment to measure performance and base compensation on reuse and rapid application assembly, rather than on volume of code produced. "You want them to build solutions, not code that has to be maintained," said one IS executive.

Companies also counsel promoting strong group dynamics. Project leaders stressed working closely together and putting trust in other team members' objects. At all costs, discourage the "not invented here" syndrome.

✓ 20. Develop Object-Oriented Metrics

At this point, reliable broad-based metrics for object development performance do not exist. Some companies are coming up with their own rules of thumb, based on a small number of projects. When developing measures of success, consider three types of metrics:

- **Predictive:** How long will the project take? What resources are required? What is the risk of failure? How great is the business impact? This metric is difficult to put in place at present because not enough object development projects have been evaluated to help companies establish a baseline.
- **Structural:** How big should a class be, with how many methods?
- **Productivity-related:** These measures range from development time, to costs, to team size, to reuse achieved. In particular, companies need new ways to measure team productivity, since lines of code generated is no longer an accurate yardstick.

Companies advised seeking guidance from object technology vendors, consultants and industry associations that are likely to have participated in enough projects to begin developing realistic benchmarks.

rewrites, and work hand-in-hand with end-users to develop complex applications that are as timely as they are useful. And, they point to the productivity benefits beginning to accrue from design and code reuse, which they anticipate will only increase as they gain experience and skill with object technology.

Taligent is committed to publicly sharing these learnings, and to using this knowledge to continue shaping its business and product plans.

Is the object world perfect yet? No. But these companies unanimously affirm that the benefits reaped far exceed the risks and obstacles they have encountered as object technology pioneers. What's more, they are confident of extensive benefits yet to come. As one CIO eloquently stated, "Object technology offers an ideal product and service creation environment that can, for the first time, keep pace with an ever-changing business."

Object-Oriented Metrics

Predictive	Structural	Productivity
<ul style="list-style-type: none"> • Project length • Resources required • Risk of failure • Business impact 	<ul style="list-style-type: none"> • Optimal class size • Appropriate number of methods • Interface standards 	<ul style="list-style-type: none"> • Development time • Project costs • Team size • Reuse achieved

A final word to the wise: However difficult it may seem, don't shirk the responsibility of establishing metrics. Without them, companies will never really know the impact of object technology on the business, or be able to justify to senior management the time and expense required.

Epilogue

Are the companies Taligent investigated pleased with the progress of their object-oriented development work, and with the resulting applications? Without a doubt. Participants cited the ability to quickly add functionality to systems that would have required complete

Implications For Taligent

As the reader has learned by now, a number of prominent corporate developers and system integrators are aggressively developing object-based applications, accumulating invaluable experience and insights along the way.

Taligent is committed to publicly sharing these learnings, and to using this knowledge to continue shaping the company's business and product plans.

Object technology as a critical foundation

It is clear that customers will aggressively adopt new technologies that offer tangible business benefits. Taligent believes, and these participants confirm, that object technology will be a core foundation technology of the 1990s, helping companies achieve their strategic business objectives. Object-based applications will speed products to market, enable rapid response to changing customer requirements, and ultimately empower end-users to create their own business solutions.

Taligent's commitment to customers

For Taligent, this study reinforced that customers are highly knowledgeable and advanced in their use of the technology. With the benefit of in-depth exposure to these real-world experiences, Taligent more fully understands these companies' requirements and priorities, and can begin to crystallize how customers will eventually use its products. This study, together with its predecessor, *A Study of America's Top Corporate Innovators*, provides a foundation for Taligent's understanding of what innovative companies have accomplished, and hope to accomplish, with object technology.

These learnings, coupled with Taligent's own object technology expertise, will uniquely enable Taligent to address many of these early adopters' challenges. Whether it's providing a world-class development environment to improve programmer productivity, making extensive use of frameworks• to facilitate reuse, or opening up the system for third party enhancement – Taligent is dedicated to delivering products that will help companies realize the full promise of object technology.

Taligent's commitment to object technology education and awareness

Taligent takes seriously its role as object technology educator in the marketplace. In that role, Taligent urges thoughtful study of the lessons gleaned from these object technology pioneers. They offer invaluable guidance, and stand as a model for others who are attracted to the enormous potential benefits of object technology but are hesitating over the next big step. Moving forward, Taligent will continue investigating – and sharing – the object technology experiences of the marketplace.

On a personal note...

The creators of this study, Nancy Deyo, Joe Gillach and Keith Wescourt, would like to thank these object technology leaders on behalf of Taligent and all the corporate developers and vendors who stand to benefit from the lessons you have learned. Your generosity in publicly sharing your hard-won experience will, we are confident, be instrumental in encouraging others to leverage object technology to transform their businesses. Bravo!

• A framework is the design of a set of objects that collaborate to carry out a set of responsibilities. University of Illinois tech report UIUCDCS91-1696.

© 1993 Taligent Inc. All Rights Reserved
Taligent and the Taligent logo are trademarks of Taligent, Inc.

All other products and services mentioned in this document are identified by the trademarks or registered trademarks as designated by the companies that market those products or services.



Taligent

10725 N. De Anza Boulevard
Cupertino, CA 95014-2000
408 255 2525

© Taligent, Inc. 1992, all rights reserved.
The Taligent identifier is a trademark of
Taligent, Inc. Produced in the USA.

TM3/2/93